Information Granules Filtering for Inexact Sequential Pattern Mining by Evolutionary Computation

Enrico Maiorino[1], Francesca Possemato[1], Valerio Modugno[2] and Antonello Rizzi[1]
[1]*Dipartimento di Ingegneria dell'Informazione, Elettronica e Telecomunicazioni (DIET)*
SAPIENZA University of Rome, Via Eudossiana 18, 00184, Rome, Italy
[2]*Dipartimento di Ingegneria Informatica, Automatica e Gestionale (DIAG)*
SAPIENZA University of Rome, Via Ariosto 25, 00185, Rome, Italy

Keywords: Granular Modeling, Sequence Data Mining, Inexact Sequence Matching, Frequent Subsequences Extraction, Evolutionary Computation

Abstract: Nowadays, the wide development of techniques to communicate and store information of all kinds has raised the need to find new methods to analyze and interpret big quantities of data. One of the most important problems in sequential data analysis is *frequent pattern mining*, that consists in finding frequent subsequences (patterns) in a sequence database in order to highlight and to extract interesting knowledge from the data at hand. Usually real-world data is affected by several noise sources and this makes the analysis more challenging, so that *approximate pattern matching* methods are required. A common procedure employed to identify recurrent patterns in noisy data is based on clustering algorithms relying on some edit distance between subsequences. When facing inexact mining problems, this plain approach can produce many spurious patterns due to multiple pattern matchings on the same sequence excerpt. In this paper we present a method to overcome this drawback by applying an optimization-based filter that identifies the most descriptive patterns among those found by the clustering process, able to return clusters more compact and easily interpretable. We evaluate the mining system's performances using synthetic data with variable amounts of noise, showing that the algorithm performs well in synthesizing retrieved patterns with acceptable information loss.

1 INTRODUCTION

Nowadays, sequence data mining is a very interesting field of research that is going to be central in the next years due to the growth of the so called "Big Data" challenge. Moreover, available data in different application fields consist in sequences (for example over time or space) of generic objects. Generally speaking, given a set of sequences defined over a particular domain, a data mining problem consists in searching for possible frequent subsequences (patterns), relying on inexact matching procedures. In this work we propose a possible solution for the so called *approximate subsequence mining* problem, in which we admit some noise in the matching process. As an instance, in computational biology, searching for recurrent patterns is a critical task in the study of DNA, aiming to identify some genetic mutations or to classify proteins according to some structural properties. Sometimes the process of pattern extraction returns sequences that differ from the others in a few positions. Then it is not difficult to understand that the choice of an adequate dissimilarity measure becomes a critical issue when we want to design an algorithm able to deal with this kind of problems. Handling sequences of objects is another challenging aspect, especially when the data mining task is defined over a *structured* domain of sequences. Thinking data mining algorithms as a building block of a wider system facing a classification task, a reasonable way to treat complex sequential data is to map sequences to $\mathbb{R}^d$ vectors by means of some *feature extraction* procedures in order to use classification techniques that deal with real valued vectors as input data. The Granular Computing (GrC) (Bargiela and Pedrycz, 2003) approach offers a valuable framework to fill the gap between the input sequence domain and the features space $\mathbb{R}^d$ and relies on the so-called *information granules* that play the role of *indistinguishable* features at a particular level of abstraction adopted for system description. The main objective of Granular modeling consists in finding the correct level of information granulation that best describes the input data. The problem of pattern data mining is similar to the problem of mining frequent item sets and subsequences. For this type of problem many works (Zaki, 2001) (Yan et al., 2003) describe search techniques for non-contiguous sequences of objects. For example, the first work (Agrawal and Srikant, 1995) of this sub-field of data mining is related to the analysis and prediction of consumer behaviour. In this context, a transaction consists in the sale of a set of items and a sequence is an ordered set of transactions. If we consider, for instance, a sequence of transactions $\langle a,b,a,c,a,c\rangle$ a possible non-contiguous subsequence could be $\langle a,b,c,c\rangle$. However, this approach is not ideal when the objective is to extract patterns where the contiguity of the component objects inside a sequence plays a fundamental role in the extraction of information. The computation biology community has developed a lot of methods for detecting frequent patterns that in this field are called *motifs*. Some works (Sinha and Tompa, 2003), (Pavesi et al., 2004) use Hamming distances to search for recurrent motifs in data. Other works employ suffix tree data structure (Zhu et al., 2007), suffix array to store and organize the search space (Ji and Bailey, 2007), or use a GrC framework for the extraction of frequent patterns in data (Rizzi et al., 2013). Most methods focus only on the recurrence of patterns in data without taking into account the concept of "information redundancy", or, in other words, the existence of overlapping among retrieved patterns. In this paper we present a new approximate subsequence mining algorithm called FRL-GRADIS (Filtered Reinforcement Learning-based GRanular Approach for DIscrete Sequences) aiming to reduce the information redundancy of RL-GRADIS (Rizzi et al., 2012) by executing an optimization-based refinement process on the extracted patterns. In particular, this paper introduces the following contributions:

1. our approach finds the patterns that maximize the knowledge about the process that generates the sequences;
2. we employ a dissimilarity measure that can extract patterns despite the presence of noise and possible corruptions of the patterns themselves;
3. our method can be applied on every kind of sequence of objects, given a properly defined similarity or dissimilarity function defined in the objects domain;
4. the filtering operation produces results that can be interpreted more easily by application's field experts;
5. considering this procedure as an inner module of a more complex classification system, it allows to further reduce the dimension of the feature space, thus better addressing the *curse of dimensionality* problem.

This paper consists of three parts. In the first part we provide some useful definitions and a proper notation; in the second part we present FRL-GRADIS as a two-step procedure, consisting of a subsequences extraction step and a subsequences filtering step. Finally, in the third part, we report the results obtained by applying the algorithm to synthetic data, showing a good overall performance in most cases.

2 PROBLEM DEFINITION

Let $\mathcal{D} = \{\alpha_i\}$ be a *domain* of *objects* α_i. The objects represent the atomic units of information. A sequence S is an ordered list of n objects that can be represented by the set of pairs

$$S = \{(i \to \beta_i) \mid i = 1,\dots,n;\ \beta_i \in \mathcal{D}\},$$

where the integer i is the order index of the object β_i within the sequence S. S can also be expressed with the compact notation

$$S \equiv \langle \beta_1, \beta_2, \dots, \beta_n \rangle$$

A sequence database *SDB* is a set of sequences S_i of variable lengths n_i. For example, the DNA sequence $S = \langle G,T,C,A,A,T,G,T,C\rangle$ is defined over the domain of the four amino acids $\mathcal{D} = \{A,C,G,T\}$.

A sequence $S_1 = \langle \beta'_1, \beta'_2, \dots, \beta'_{n_1}\rangle$ is a *subsequence* of a sequence $S_2 = \langle \beta''_1, \beta''_2, \dots, \beta''_{n_2}\rangle$ if $n_1 \leq n_2$ and $S_1 \subseteq S_2$. The *position* $\pi_{S_2}(S_1)$ of the subsequence S_1 with respect to the sequence S_2 corresponds to the order index of its first element (in this case the order index of the object β'_1) within the sequence S_2. The subsequence S_1 is also said to be *connected* if

$$\beta'_j = \beta''_{j+k} \quad \forall j = 1,\dots,n_1$$

where $k = \pi_{S_2}(S_1)$.
Two subsequences S_1 and S_2 of a sequence S are *overlapping* if

$$S_1 \cap S_2 \neq \emptyset.$$

In the example described above, the complete notation for the sequence $S = \langle G,T,C,A,A,T,G,T,C\rangle$ is

$$S = \{(1 \to G), (2 \to T), (3 \to C), \dots\}$$

and a possible connected subsequence $S_1 = \langle A,T,G\rangle$ corresponds to the set

$$S_1 = \{(5 \to A), (6 \to T), (7 \to G)\}.$$

Notice that the objects of the subsequence S_1 inherit the order indices from the containing sequence S, so that they are univocally referred to their original positions in S. From now on we will focus only on connected subsequences, therefore the connection property will be implicitly assumed.

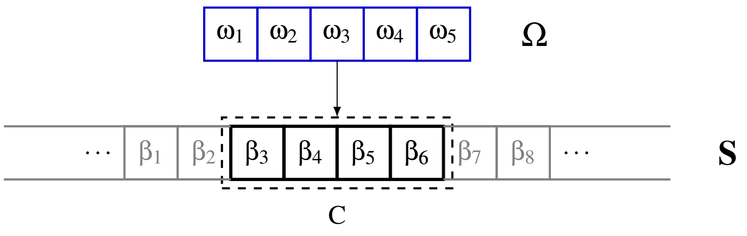

Figure 1: Coverage of the pattern Ω over the subsequence $C \subseteq S$ with tolerance δ. Black boxes and gray boxes represent respectively the covered and the uncovered objects of the sequence S. Notice that if $\delta > 0$ the sequences Ω and C need not to be of the same length.

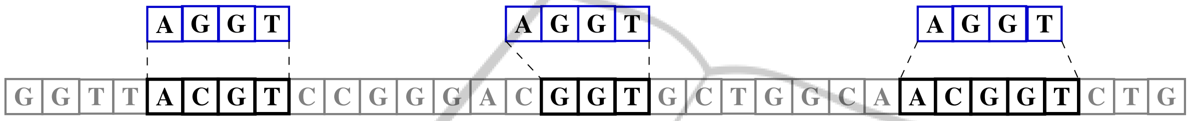
A G G T
A G G T
A G G T
G G T T A C G T C C G G G A C G G T G C T G G C A A C G G T C T G

Figure 2: Coverage examples in the case of DNA sequences. The searched pattern $\langle A,G,G,T\rangle$ is found 3 times with tolerance $\delta \leq 1$ using the Levenshtein distance. The three occurrences show all the edit operations allowed by the considered edit distance, respectively objects substitution, deletion and insertion.

2.1 Pattern *Coverage*

The objective of this algorithm is to find a set of frequent subsequences of objects named as *patterns*. A pattern Ω is a subsequence of objects $\langle \omega_1, \omega_2, ..., \omega_{|\Omega|}\rangle$, with $\omega_i \in \mathcal{D}$, that is more likely to occur within the dataset SDB. Patterns are unknown *a priori* and represent the underlying information of the dataset records. Moreover, each sequence is subject to noise whose effects include the addition, substitution and deletion of objects in a random uncorrelated fashion and this makes the recognition of recurrent subsequences more challenging.

Given a sequence $S \in SDB$ and a set of patterns $\Gamma = \{\Omega_1, ..., \Omega_m\}$, we want to determine a quality criterion for the description of S in terms of the pattern set Γ. A connected subsequence $C \subseteq S$ is said to be **covered** by a pattern $\Omega \in \Gamma$ iff $d(C,\Omega) \leq \delta$, where $d(\cdot,\cdot)$ is a properly defined distance function and δ is a fixed tolerance (Fig. 1). The **coverage** $\mathcal{C}_{\Omega}^{(\delta)}(S)$ of the pattern Ω over the sequence S is the union set of all non-overlapping connected subsequences covered by the pattern. We can write,

$$\mathcal{C}_{\Omega}^{(\delta)}(S) = \bigcup_i \Big[C_i \subseteq S \text{ s.t.} \\ d(C_i,\Omega) \leq \delta \wedge C_i \cap C_j = \emptyset \, \forall \, i \neq j \Big]. \quad (1)$$

Formally, this set is still not well defined until we expand on the meaning of the property

$$C_i \cap C_j = \emptyset, \quad (2)$$

which is the requirement for the covered subsequences to be non-overlapping. Indeed, we need to include additional rules on how to deal with these overlappings when they occur. To understand better, let us recall the example of the DNA sequences presented above, where the dissimilarity measure between two sequences is the Levenshtein distance. The set of all covered subsequences C_i (in this context referred to as *candidates*) by the pattern Ω over the sequence S will consist only of sequences with values of length between $|\Omega| - \delta$ and $|\Omega| + \delta$. Indeed, these bounds correspond respectively to the extreme cases of deleting and adding δ objects to the subsequence. In case of two overlapping candidates C_i and C_j, in order to satisfy the property (2) of the coverage $\mathcal{C}_{\Omega}^{(\delta)}(S)$, we have to define a rule to decide which subsequence belongs to the set $\mathcal{C}_{\Omega}^{(\delta)}(S)$ and which does not. Candidates with smaller distances from the searched pattern Ω are chosen over overlapping candidates with higher distances. If the two overlapping candidates have the same distance the first starting from the left is chosen, but if also their starting position is the same the shorter one (i.e. smaller length value) has the precedence.

A coverage example in the context of the DNA sequences is shown in Fig. 2. The coverage of the pattern $\Omega = \langle A,G,G,T\rangle$ over the sequence S is $\mathcal{C}_{\Omega}^{(\delta)}(S) = \langle A,C,G,T\rangle \cup \langle G,G,T\rangle \cup \langle A,C,G,G,T\rangle$.

Similarly, the compound coverage of the pattern set Γ is defined as

$$\mathcal{C}_{\Gamma}^{(\delta)}(S) = \bigcup_{\Omega \in \Gamma} \mathcal{C}_{\Omega}^{(\delta)}(S). \quad (3)$$

It is important to notice that, in this case, this set can include overlapping subsequences only if they be-

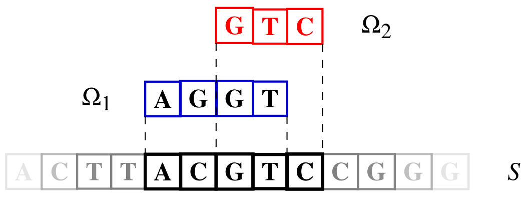

Figure 3: Example of the compound coverage of multiple symbols, where the symbols $\langle G,T,C\rangle$ and $\langle A,G,G,T\rangle$ have Levenshtein distances from the corresponding subsequences equal to 0 and 1, respectively. Notice that different symbols can cover overlapping subsequences, while competing coverages of the same symbol are not allowed and only the most similar subsequence is chosen.

long to coverages of different patterns (i.e. it is assumed that different patterns can overlap). For example consider the case shown in Fig. 3. The coverage $C^{(\delta)}_{\{\Omega_1,\Omega_2\}}(S)$ for the patterns $\Omega_1 = \langle A,G,G,T\rangle$ and $\Omega_2 = \langle G,T,C\rangle$ is equal to $C^{(\delta)}_{\{\Omega_1,\Omega_2\}}(S) = \langle A,G,G,T,C\rangle$.

3 THE MINING ALGORITHM

In this section, we describe FRL-GRADIS, as a clustering-based sequence mining algorithm. It is able to discover clusters of connected subsequences of variable lengths that are frequent in a sequence dataset, using an inexact matching procedure. FRL-GRADIS consists in two main steps:

- the *Symbols Alphabet Extraction*, which addresses the problem of finding the most frequent subsequences within a SDB. It is performed by means of the clustering algorithm RL-GRADIS (Rizzi et al., 2012) that identifies frequent subsequences as representatives of dense clusters of similar subsequences. These representatives are referred to as *symbols* and the pattern set as the *alphabet*. The clustering procedure relies on a properly defined edit distance between the subsequences (e.g. Levenshtein distance, DTW, etc..). However, this approach alone has the drawback of extracting many superfluous symbols which generally dilute the pattern set and deteriorate the *interpretability* of the produced pattern set.
- the *Alphabet Filtering* step deals with the problem stated above. The objective is to filter out all the spurious or redundant symbols contained in the alphabet produced by the symbols extraction step. To accomplish this goal we employ a heuristic approach based on evolutionary optimization over a validation SDB.

One of the distinctive features of this algorithm is its generality with respect to the kind of data contained in the input sequence database (e.g., sequences of real numbers or characters as well as sequences of complex data structures). Indeed, both steps outlined above take advantage of a dissimilarity-based approach, with the dissimilarity function being a whatever complex measure between two ordered sequences, not necessarily metric.

In the following, we first describe the main aspects of the symbols alphabet extraction procedure, then we present the new filtering method. For more details on the symbols alphabet construction we refer the reader to (Rizzi et al., 2012).

3.1 Frequent Subsequences Identification

Consider the input training dataset of sequences $\mathcal{T} = \{S_1, S_2, ..., S_{|\mathcal{T}|}\}$ and a properly defined dissimilarity measure $d : \mathcal{T} \times \mathcal{T} \to \mathbb{R}$ between two objects of the training dataset (e.g., Levenshtein distance for strings of characters). The goal of the subsequences extraction step is the identification of a finite set of symbols $\mathcal{A}_e = \{\Omega_1, \Omega_2, ..., \Omega_{|\mathcal{A}_e|}\}$,[1] computed using the distance $d(\cdot,\cdot)$ in a free clustering procedure. The algorithm we chose to accomplish this task is RL-GRADIS which is based on the well-known Basic Sequential Algorithmic Scheme (BSAS) clustering algorithm (Rizzi et al., 2012). Symbols are found by analysing a suited set of variable-length subsequences of $\mathcal{T}$, also called n-grams, that are generated by expanding each input sequence $S \in \mathcal{T}$. The expansion is done by listing all n-grams with lengths varying between the values $l_{\min}$ and $l_{\max}$. The parameters $l_{\min}$ and $l_{\max}$ are user-defined and are respectively the minimum and maximum admissible length for the mined patterns. The extracted n-grams are then collected into the SDB $\mathcal{N}$. At this point, the clustering procedure is executed on $\mathcal{N}$. For each cluster we compute its representative, defined by the Minimum Sum of Distances (MinSOD) technique (Rizzi et al., 2012), as the element having the minimum total distance from the other elements of the cluster. This technique allows to represent the corresponding clusters by means of their most characteristic elements.

The quality of each cluster is measured by its *firing strength* f, where $f \in [0,1]$. Firing strengths are used to track the dynamics describing the *updating rate* of the clusters when the input stream of subsequences $\mathcal{N}$ is analyzed. A reinforcement learning procedure is used to dynamically update the list

[1]The subscript “e” stands for “extraction” as in extraction step.

of candidate symbols based on their firing strength. Clusters with a low rate of update (low firing strength) are discarded in an on-line fashion, along with the processing of the input data stream $\mathcal{N}$. RL-GRADIS maintains a dynamic list of candidate symbols, named *receptors*, which are the representatives of the active clusters. Each receptor's firing strength (i.e. the firing strength of its corresponding cluster) is dynamically updated by means of two additional parameters, $\alpha, \beta \in [0,1]$. The α parameter is used as a *reinforcement weight* factor each time a cluster $\mathcal{R}$ is updated, i.e., each time a new input subsequence is added to $\mathcal{R}$. The firing strength update rule is defined as follows:

$$f(\mathcal{R}) \leftarrow f(\mathcal{R}) + \alpha(1 - f(\mathcal{R})). \tag{4}$$

The β parameter, instead, is used to model the speed of *forgetfulness* of receptors according to the following formula:

$$f(\mathcal{R}) \leftarrow (1-\beta) f(\mathcal{R}). \tag{5}$$

The firing strength updating rules shown in Eqs. (4) and 5 are performed for each currently identified receptor, soon after the analysis of each input subsequence. Therefore, receptors/clusters that are not updated frequently during the analysis of $\mathcal{N}$ will likely have a low strength value and this will cause the system to remove the receptor from the list.

3.2 Subsequences Filtering

As introduced above, the output alphabet $\mathcal{A}_e$ of the clustering procedure is generally redundant and includes many spurious symbols that make the recognition of the true alphabet quite difficult.

To deal with this problem, an optimization step is performed to reduce the alphabet size, aiming at retaining only the most significant symbols, i.e. only those that best resemble the original, unknown ones. Since this procedure works like a filter, we call the output of this optimization the *filtered alphabet* $\mathcal{A}_f$ and, clearly, $\mathcal{A}_f \subset \mathcal{A}_e$ holds. Nevertheless, it is important for the filtered alphabet's size not to be smaller than the size of the true alphabet, since in this case useful information will be lost. Let $\Gamma \subset \mathcal{A}_e$ be a candidate subset of symbols of the alphabet $\mathcal{A}_e$ and $S \in \mathcal{V}$ a sequence of a *validation SDB* $\mathcal{V}$. We assume the *descriptive power* of the symbols set Γ, with respect to the sequence S, to be proportional to the quantity $|\mathcal{C}_\Gamma^{(\delta)}(S)|$ (cfr Eq. (3)), i.e. the number of objects $\beta_i \in S$ covered by the symbols set Γ. In fact, intuitively, a lower number of uncovered objects in the whole SDB by Γ symbols can be considered as a clue that Γ itself will likely contain the true alphabet. The normalized number of uncovered objects in a sequence S by a pattern set Γ corresponds to the quantity

$$P = \frac{|S| - |\mathcal{C}_\Gamma^{(\delta)}(S)|}{|S|}, \tag{6}$$

where the operator $|\cdot|$ stands for the cardinality of the set. The term P assumes the value 0 when the sequence S is completely covered by the pattern set Γ and the value 1 when none of the symbols in Γ are present in the sequence S. Notice that $\mathcal{C}_\Gamma^{(\delta)}(S)$ depends on the parameter δ which represents the tolerance of the system towards the corruption of symbols' occurrences caused by noise.

On the other hand, a bigger pattern set is more likely to contain spurious patterns which tend to hinder the interpretability of the obtained results, so smaller set sizes are to be preferred. This property can be described with the normalized alphabet size

$$Q = \frac{|\Gamma|}{|\mathcal{A}_e|}, \tag{7}$$

where $\mathcal{A}_e$ is the alphabet of symbols extracted by the clustering procedure described in the last section. Clearly, the cardinality of $\mathcal{A}_e$ represents an upper bound for the size of the filtered alphabet, so the term Q ranges from 0 to 1. The terms P and Q generally show opposite trends, since a bigger set of symbols is more likely to cover a bigger portion of the sequence and vice versa.

Finding a tradeoff between these two quantities corresponds to minimizing the convex objective function

$$G_S^{(\delta)}(\Gamma) = \lambda Q + (1-\lambda) P \tag{8}$$

where $\lambda \in [0,1]$ is a meta-parameter that weighs the relative importance between the two constributions. It is easy to verify that

$$0 \leq G_S^{(\delta)}(\Gamma) \leq 1. \tag{9}$$

More generally, for a validation SDB $\mathcal{V}$, the global objective function is the mean value of $G_S^{(\delta)}(\Gamma)$ over all sequences $S_i \in \mathcal{V}$, hence

$$G_{\mathcal{V}}^{(\delta)}(\Gamma) = \frac{\sum\limits_{1 \leq i \leq |\mathcal{V}|} G_{S_i}^{(\delta)}(\Gamma)}{|\mathcal{V}|} \tag{10}$$

and the best symbols set after the optimization procedure is

$$\mathcal{A}_f = \underset{\Gamma \subset \mathcal{A}_e}{\operatorname{argmin}}\, G_S^{(\delta)}(\Gamma). \tag{11}$$

To solve the optimization problem described by Eq. (11) we employ a standard genetic algorithm, where each individual of the population is a subset Γ of the extracted alphabet $\mathcal{A}_e = \{\Omega_1, ..., \Omega_{|\mathcal{A}_e|}\}$. The genetic code of the individual is encoded as a binary sequence E of length $|\mathcal{A}_e|$ of the form

$$E_\Gamma = \langle e_1, e_2, ..., e_{|\mathcal{A}_e|} \rangle \tag{12}$$

with

$$e_i = \begin{cases} 1 & \text{iff } \Omega_i \in \Gamma \\ 0 & \text{otherwise} \end{cases}.$$

It is important not to mistake genetic codes with the SDB sequences described earlier, even if they are both formally defined as ordered sequences.

Given a validation dataset $\mathcal{V}$ and a fixed tolerance δ, the fitness value $F(E_\Gamma)$ of each individual E_Γ is inversely proportional to the value of the objective function introduced in the last paragraph, hence

$$F(E_\Gamma) = 1 - G_{\mathcal{V}}^{(\delta)}(\Gamma) \tag{13}$$

The computation is then performed with standard crossover and mutation operators between the binary sequences and the stop condition is met when the maximum fitness does not change for a fixed number N_{stall} of generations or after a given maximum number N_{max} of iterations. When the evolution stops, the filtered alphabet $\mathcal{A}_f = \widetilde{\Gamma}$ is returned, where $\widetilde{\Gamma}$ is the symbols subset corresponding to the fittest individual $E_{\widetilde{\Gamma}}$.

4 TESTS AND RESULTS

In this section, we present results from different experiments that we designed to test the effectiveness and performance of FRL-GRADIS in facing problems with varying complexity.

4.1 Data Generation

We tested the capabilities of FRL-GRADIS on synthetic sequence databases composed of textual strings. For this reason, the domain of the problem is the English alphabet

$$\mathcal{D} = \{A, B, C, ..., Z\}.$$

Modelled noise consists of random characters insertions, deletions and substitutions to the original string. For this reason a natural choice of dissimilarity measure between sequences is the Levenshtein distance, that measures the minimum number of edit steps necessary to transform one string of characters into another. The dataset generator works as follows:

1. the true symbols alphabet $\mathcal{A}_\text{t}$ is generated. This alphabet consists of N_{sym} symbols with lengths normally distributed around the mean value L_{sym}. Each character is chosen in $\mathcal{D}$ with uniform probability and repeated characters are allowed;
2. a training SDB $\mathcal{T}$ and a validation SDB $\mathcal{V}$ respectively composed of N_{tr} and N_{val} sequences are generated. Each of these sequences is built by concatenating N_{symseq} symbols chosen randomly from $\mathcal{A}_\text{t}$. Notice that generally $N_{\text{symseq}} > N_{\text{sym}}$ so there will be repeated symbols;
3. in each sequence, every symbol will be subject to noise with probability μ. The application of noise to a symbol in a sequence corresponds to the deletion, substitution or insertion of one character to that single instance of the symbol. This kind of noise is referred to as *intra-pattern* noise;
4. a user-defined quantity of random characters is added between instances of symbols in each sequence. This noise is called *inter-pattern* noise. Such quantity depends on the parameter η that corresponds to the ratio between the number of characters belonging to actual symbols and the total number of character of the sequence after the application of inter-pattern noise, that is,

   $$\eta = \frac{(\text{\# symbol characters})}{(\text{\# total characters})}.$$

   Notice that the amount of inter-pattern noise is inversely proportional to the value of η.

The generated datasets $\mathcal{T}$ and $\mathcal{V}$ are then ready to be used as input of the FRL-GRADIS procedure. Notice that the true alphabet $\mathcal{A}_\text{t}$ is unknown in real-world applications and here is used only to quantify the performance of the algorithm.

4.2 Quality Measures

We now introduce the quality measures used in the following tests to evaluate the mining capabilities of the FRL-GRADIS algorithm. These measures are computed for the resulting alphabets obtained from both the extraction and the filtering steps presented in Section 3, in order to highlight the improvement made by the filtering procedure (i.e. the improvement of FRL-GRADIS over RL-GRADIS).

The *redundance* R corresponds to the ratio between the cardinality of the alphabet $\mathcal{A}$ and the true alphabet $\mathcal{A}_\text{t}$, that is,

$$R = \frac{|\mathcal{A}|}{|\mathcal{A}_\text{t}|} \tag{14}$$

Clearly, since the filtering step selects a subset $\mathcal{A}_{\text{f}}$ (filtered alphabet) of the extracted alphabet $\mathcal{A}_{\text{e}}$, we always have that

$$R_{\text{f}} < R_{\text{e}}.$$

The redundance measures the amount of unnecessary symbols that are found by a frequent pattern mining procedure and it ranges from zero to infinite. When $R > 1$ some redundant symbols have been erroneously included in the alphabet, while when $R < 1$ some have been missed, the ideal value being $R = 1$.

It is important to notice that the redundancy depends only on the number of symbols reconstructed, but not by their similarity with respect to the original alphabet. For this purpose we also introduce the *mining error* E, defined as the mean distance between each symbol Ω_i of the true alphabet $\mathcal{A}_{\text{t}}$ and its best match within the alphabet $\mathcal{A}$, where the best match means the symbol with the least distance from Ω_i. In other words, considering $\mathcal{A}_{\text{t}} = \{\Omega_1, \ldots, \Omega_{|\mathcal{A}_{\text{t}}|}\}$ and $\mathcal{A} = \{\tilde{\Omega}_1, \ldots, \tilde{\Omega}_{|\mathcal{A}|}\}$, the mining error corresponds to

$$E = \frac{\sum_i d(\Omega_i, \tilde{\Omega}_{(i)})}{|\mathcal{A}_{\text{t}}|} \tag{15}$$

where

$$\tilde{\Omega}_{(i)} = \underset{\tilde{\Omega} \in \mathcal{A}}{\text{argmin}}\ d(\Omega_i, \tilde{\Omega}).$$

This quantity has the opposite role of the redundancy, in fact it keeps track of the general accuracy of reconstruction of the true symbols regardless of the generated alphabet size. It assumes non-negative values and the ideal value is 0. For the same reasons stated above the inequality

$$E_{\text{f}} \geq E_{\text{e}}$$

holds, so the extraction procedure's mining error constitutes a lower bound for the mining error obtainable with the filtering step.

4.3 Results

We executed the algorithm multiple times for different values of the noise parameters, to assess the different response of FRL-GRADIS to increasing amounts of noise. Most parameters have been held fixed for all the tests and they are listed in table 1.

We present the results of tests performed with $\mu = 0.5$ and variable amounts of inter-pattern noise η. It means that about half of the symbols in a sequence are subject to the alteration of one character and increasing amounts of random characters are added between symbols in each sequence. The results obtained with this configuration are shown in figs. 4 and 5.

Table 1: Fixed parameters adopted for the tests. The parameter δ corresponds to the tolerance of the Levenshtein distance considered when calculating the coverage as in Eq. (1) while λ weighs the two terms of the objective function of Eq. (8). The values shown in the second part of the table refer to the genetic algorithm's parameters. N_{pop} corresponds to the population size, N_{elite} is the fraction of individuals who are guaranteed to survive and be copied to the new population in each iteration, p_{cross} and $p_textmut$ are respectively the crossover and mutation probabilities. The evolution terminates if N_{evol} iterations have been performed or if for a number N_{stall} of iterations the maximum fitness has not changed.

Parameter	Value	Parameter	Value
N_{tr}	50	N_{val}	25
N_{sym}	5	N_{symseq}	10
l_{min}	4	l_{max}	12
δ	1	λ	0.5
N_{pop}	100	N_{elite}	0.1
p_{cross}	0.8	p_{mut}	0.3
N_{max}	100	N_{stall}	50

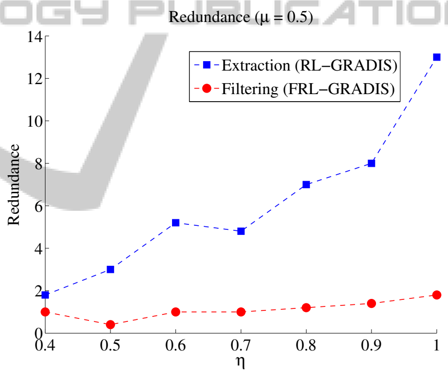

Figure 4: Plot of the redundance R of the extraction (RL-GRADIS) and filtering (FRL-GRADIS) steps for $\mu = 0.5$.

The redundancy plot in fig. 4 shows an apparently paradoxical trend of the extraction procedure's redundancy: with decreasing amounts of inter-pattern noise (i.e. increasing values of η) the extraction algorithm performs more poorly, leading to higher redundancies. That can be easily explainable by recalling how the clustering procedure works.

Higher amounts of inter-pattern noise mean that the frequent symbols are more likely to be separated by random strings of characters. These strings of uncorrelated characters generate very sparse clusters with negligible cardinality that are very likely to be deleted during the clustering's reinforcement step. Clusters corresponding to actual symbols, instead, are more active and compact, their bounds being clearly defined by the noise characters, and so they are more likely to survive the reinforcement step.

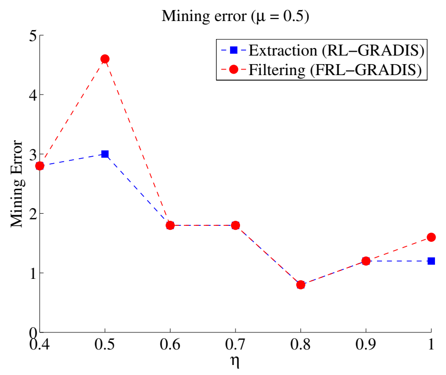

Figure 5: Plot of the mining error E of the extraction (RL-GRADIS) and filtering steps (FRL-GRADIS) for $\mu = 0.5$.

In case of negligible (or non-existent) inter-pattern noise, instead, different symbols are more likely to occur in frequent successions that cause the generation of many clusters corresponding to spurious symbols, obtained from the concatenation of parts of different symbols. The filtering procedure overcomes this inconvenience, as it can be seen from fig. 4 that it is nearly not affected by the amount of inter-pattern noise. As it is evident, the filtering procedure becomes fundamental for higher values of the parameter η, where the clustering produces highly redundant alphabets that would be infeasible to handle in a real-world application. Fig. 5 shows that the mining error after the filtering procedure remains mostly the same for all values of η, which means that the system is robust to the moderate alteration of the input signal.

In general, we can conclude that the system allows for a remarkable synthesis of the extracted alphabet despite of a modest additional mining error.

5 CONCLUSIONS

In this work we have presented a new approach to sequence data mining, focused on improving the interpretability of the frequent patterns found in the data. For this reason, we employed a two-steps procedure composed of a clustering algorithm, that extracts the frequent subsequences in a sequence database, and a genetic algorithm that filters the returned set to retrieve a smaller set of patterns that best describes the input data. For this purpose we introduced the concept of coverage, that helps in recognizing the true presence of a pattern within a sequence affected by noise. The results show a good overall performance and lay the foundations for further tests and improvements.

REFERENCES

Agrawal, R. and Srikant, R. (1995). Mining sequential patterns. In *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*, pages 3–14. IEEE.

Bargiela, A. and Pedrycz, W. (2003). *Granular computing: an introduction*. Springer.

Ji, X. and Bailey, J. (2007). An efficient technique for mining approximately frequent substring patterns. In *Data mining workshops, 2007. ICDM workshops 2007. Seventh IEEE international conference on*, pages 325–330. IEEE.

Pavesi, G., Mereghetti, P., Mauri, G., and Pesole, G. (2004). Weeder web: discovery of transcription factor binding sites in a set of sequences from co-regulated genes. *Nucleic acids research*, 32(suppl 2):W199–W203.

Rizzi, A., Del Vescovo, G., Livi, L., and Frattale Mascioli, F. M. (2012). A New Granular Computing Approach for Sequences Representation and Classification. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pages 2268–2275.

Rizzi, A., Possemato, F., Livi, L., Sebastiani, A., Giuliani, A., and Mascioli, F. M. F. (2013). A dissimilarity-based classifier for generalized sequences by a granular computing approach. In *IJCNN*, pages 1–8. IEEE.

Sinha, S. and Tompa, M. (2003). Ymf: a program for discovery of novel transcription factor binding sites by statistical overrepresentation. *Nucleic acids research*, 31(13):3586–3588.

Yan, X., Han, J., and Afshar, R. (2003). Clospan: Mining closed sequential patterns in large datasets. In *Proceedings of SIAM International Conference on Data Mining*, pages 166–177. SIAM.

Zaki, M. J. (2001). Spade: An efficient algorithm for mining frequent sequences. *Machine learning*, 42(1-2):31–60.

Zhu, F., Yan, X., Han, J., and Yu, P. S. (2007). Efficient discovery of frequent approximate sequential patterns. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 751–756. IEEE.